Universidad de las Ciencias Informáticas

Título: Serere 2.0. Instalador del sistema operativo GNU/Linux Nova.

Autor: Yunier Soler Franco.

Tutores: Ing. Raydel Miranda Gómez, Ing. Allan Pierra Fuentes.

Presentado en opción del Título de Ingeniero en Ciencias Informáticas

Ciudad de la Habana, Cuba

Mayo, 2010

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año ____.

Yunier Soler Franco	Ing. Raydel Miranda Gómez	Ing. Allan Pierra Fuentes
_____	_____	_____
Firma del autor	Firma del tutor	Firma del tutor

Agradecimientos:

A mis tutores, a mis compañeros de trabajo, por haberme ayudado en mi formación profesional y por hacerme sentir en familia durante cada día de trabajo. A todos los que han participado en el desarrollo de este trabajo.

Dedicatoria

A mis padres que han apoyado mis estudios toda la vida.

A mi novia que me ha acompañado cada día de mi carrera.

A mi familia y amigos que son muchos.

Resumen

En esta investigación se presenta el proceso de desarrollo de una nueva versión del instalador del sistema operativo GNU/Linux Nova, como una solución independiente de las líneas de desarrollo de la distribución. Se muestra el uso de la metodología ágil conocida como Programación Extrema y la integración del lenguaje de programación ,herramientas de desarrollo, pruebas unitarias, control de versiones y herramientas de gestión como entorno de desarrollo. Este trabajo constituye un aporte al desarrollo de los instaladores de sistemas operativos GNU/Linux.

PALABRAS CLAVE

Instalador, sistema operativo, solución independiente, integración

Tabla de Contenidos

Introducción.

Cuba se encuentra actualmente en un proceso de informatización de la sociedad, llevando a todos los rincones del país las tecnologías de la información y el conocimiento. Paralelamente se trabaja para eliminar la dependencia tecnológica que actualmente afecta en gran medida a la economía nacional. Para cumplir el objetivo de acercar a Cuba a la soberanía tecnológica se ha planteado la estrategia de migrar las plataformas, herramientas y servicios informáticos de todas las empresas e instituciones cubanas a tecnologías de software libre y código abierto.

Una institución líder en el proceso de migración es la Universidad de las Ciencias Informáticas (UCI), que cuenta con una facultad que estudia y desarrolla estas tecnologías. Una de las mayores dificultades que se enfrenta es que la gran mayoría de los ordenadores utilizan como sistema operativo a Microsoft Windows en sus distintas versiones. Este sistema operativo, desarrollado por la empresa norteamericana Microsoft, suma a sus limitantes de software privativo, las asociadas al bloqueo económico impuesto por los Estados Unidos. Por tal motivo, es un punto clave en la migración contar con un sistema operativo basado en tecnologías libres que sirva de base para el resto del proceso.

En el mundo existen sistemas operativos libres como las muchas distribuciones de GNU/Linux entre las cuales se encuentra Nova, distribución cubana de GNU/Linux desarrollada por la UCI, la cual ha sido seleccionada para llevar a cabo el proceso de migración nacional.

Nova es un sistema operativo construido principalmente en base a tecnologías libres, basado inicialmente en la distribución GNU/Linux Gentoo, es el fruto del trabajo de un grupo de estudiantes y profesores, principalmente de la Facultad 10 de la UCI, organizados en un proyecto productivo que cuentan ya con una versión estable llamada Nova Baire liberada en el marco de la XIII edición de la Convención y Feria Internacional Informática 2009 con sede en la Habana, en la cual se obtuvo una Mención Especial.

A partir de los resultados de la presentación de Nova en este evento se comenzó a trabajar en lo que sería la versión 2.0 de la distribución y se reorganizó el equipo de desarrollo para trabajar en diferentes líneas de trabajo a partir de las necesidades de las instituciones del país, como son, Nova Escritorio para ordenadores personales, Nova Servidores para el trabajo con los servidores, Nova Clientes Ligeros para las computadoras sin disco duro y otras.

Entre los productos que se desarrollan para Nova está su instalador, que recibe el nombre de Serere y su función guiar al usuario en el proceso de instalación y configuración del sistema operativo en el ordenador. A pesar de que el proceso actual de instalación de Nova cumple las condiciones básicas para instalar un sistema operativo, no se adapta a las condiciones reales del proyecto y a las necesidades del país.

Actualmente, las diferentes líneas de desarrollo de Nova deben incluir las librerías y herramientas que Serere necesita para su correcto funcionamiento, muchas de las cuales no son necesarias para el sistema. Además, el proceso de instalación no permite personalizar el sistema en cuanto a aplicaciones se refiere e interactúa poco con el hardware a la hora de obtener información del mismo para configurar el sistema, creando insatisfacciones a los usuarios e incompatibilidad del sistema instalado con el hardware. Esta **situación problemática** permite plantearse el siguiente **problema científico**: ¿Cómo lograr una solución de instalación más completa e independiente de las líneas de desarrollo de Nova?

Como **objeto de estudio** para esta investigación fueron seleccionados los instaladores de las distribuciones de GNU/Linux más exitosos. Las experiencias y conocimientos adquiridos en esta investigación serán aplicados en el **campo de acción** limitado por el instalador de GNU/Linux Nova.

Se trazó como **objetivo general** desarrollar Serere 2.0.

El objetivo general se divide en los siguientes **objetivos específicos**:

1. Implementar la aplicación para la instalación de GNU/Linux Nova.
2. Construir un sistema GNU/Linux que incluya únicamente las librerías y herramientas necesarias para dicha aplicación.

Se plantea como **idea a defender** en esta investigación que un instalador para el sistema operativo GNU/Linux Nova que cuente con las siguientes características:

- Independencia del sistema a instalar.
- Capacidad de instalar paquetes adicionales.
- Gestor de particiones propio.
- Capacidad de detectar información del hardware.

aumentará la compatibilidad con el hardware y la aceptación de los usuarios. Además exonerará a los desarrolladores de las distintas líneas de tener que incluir las librerías y dependencias necesarias para su funcionamiento.

El proceso de investigación estuvo dirigido por las siguientes **tareas**:

1. Investigación sobre el funcionamiento de los instaladores más usados.
2. Implementación de vistas para la aplicación de instalación de GNU/Linux Nova.
3. Investigación sobre la construcción de sistemas operativos GNU/Linux.
4. Construcción de un sistema operativo GNU/Linux.

El **método** utilizado fue: **analítico-sintético**, que permitió descomponer las características de los instaladores más usados de forma tal que después pudieron ser identificadas las fortalezas de estos y pudieron ser aplicadas a otros instaladores.

Capítulo 1: Fundamentación del tema.

Antecedentes del Instalador de Nova.

La necesidad de un instalador para Nova fue detectada desde los mismos inicios del desarrollo de la distribución, por lo que se decidió estudiar la posibilidad de utilizar alguno de los instaladores de otras distribuciones de GNU/Linux o desarrollar uno. Este estudio arrojó como resultado que se debía desarrollar un instalador, pues los existentes, eran y son en la actualidad inconsistentes con los propósitos del proyecto Nova. Así surgió Serere, que ha acompañado a Nova, hasta llegar a su versión 1.3. El proceso de instalación y configuración está compuesto por las siguientes vistas **(ver Anexo 1)**.

Selección de modo de instalación.

En esta vista se le permite al usuario seleccionar el modo en que va a instalar el sistema ya sea de forma *automática* o *manual.* El modo automático le evita al usuario con poca experiencia instalando sistemas GNU/Linux, tener que lidiar con configuraciones para las cuales se debe tener un conocimiento más avanzado en el tema. El modo manual, recomendado para usuarios con experiencia, permite configurar opciones avanzadas para la instalación y configuración del sistema.

Selección de zona horaria.

En esta vista se muestra un mapa mundial con las diferentes zonas horarias , lo cual hace posible que el usuario de forma sencilla seleccione la zona horaria en la que se encuentra ubicado.

Selección de punto de montaje.

En esta vista el usuario puede seleccionar en que disco o partición va a instalar la raíz principal de el sistema, además de posibilitar la gestión de discos y particiones mediante la herramienta Gparted.

Configuración de red.

En esta vista el usuario puede configurar las interfaces de red de su ordenador ya sea utilizando DHCP[1] o de forma manual. Esta configuración será utilizada por el sistema después de ser instalado.

Configuración de usuarios.

En esta vista se da la posibilidad de configurar los distintos usuarios que tendrá el sistema después de ser instalado. Se podrán añadir varios usuarios, además del usuario **root**[2], asignándole a cada uno los permisos que tendrán a la hora de interactuar con el sistema.

Proceso de instalación.

Esta vista solo muestra información al usuario del estado en que se encuentra el proceso de instalación, aprovechando el tiempo que demora el proceso para mostrar algunos datos de interés para el usuario.

Estas características son comunes en la mayoría de los instaladores de sistemas operativos y en Serere 1.3 permiten instalar Nova sin dificultad. Llegado este punto es necesario revisar las funcionalidades de Serere con el propósito de mejorarlas e incluir otras que puedan contribuir al desarrollo y la aceptación de Nova.

En el caso de la selección del modo de instalación, una opción que permita utilizar el disco duro completo y otra que permita personalizar el particionado, no son suficientes ya que un usuario con poca o ninguna experiencia en este tipo de gestiones puede perder información de sus discos involuntariamente. El proceso de gestión de particiones requiere alguna experiencia por parte de los usuarios en cualquier sistema operativo, pero debido a que se utiliza la herramienta Gparted no es posible para el equipo de desarrollo de Serere intentar facilitar o hacer más intuitivo este proceso.

---

[1] *Siglas en inglés de Dynamic Host Configuration Protocol – Protocolo de Configuración Dinámica de Servidor.*

[2] *Usuario administrador en sistemas GNU/Linux.*

Una limitación que presenta la versión actual del instalador es que no permite seleccionar el idioma o la distribución de teclado que desee el usuario para usar en el sistema operativo instalado. Los archivos de configuración no son manejados de la forma más correcta y en ocasiones algunos son reemplazados por archivos predefinidos creando algunas dificultades en equipos diferentes a los utilizados para construir el sistema.

Instaladores de distribuciones exitosas.

En la actualidad entre las distribuciones de GNU/Linux más exitosas y aceptadas por los usuarios se encuentran Ubuntu, OpenSUSE, CentOS y Fedora. El estudio del funcionamiento de los instaladores de estas distribuciones puede arrojar datos interesantes para el equipo de desarrollo de Serere. A continuación se describen las características principales de los mismos, haciendo énfasis en las funcionalidades con las que no cuenta el instalador de Nova.**(ver Anexo 1)**

Ubiquity.

Es el nombre del instalador de Ubuntu que ha acompañado a esta distribución hasta su versión actual (Ubuntu Karmic 9.10, en el momento que se escribe este documento). El estudio realizado sobre el funcionamiento del mismo permitió observar sus características distintivas y detectar qué funcionalidades pudieran ser incluidas en el desarrollo de Serere 2.0.

Desarrollo y funcionamiento.

Ubiquity cuenta con diferentes interfaces para cada uno de sus productos (Ubuntu, Kubuntu, Edubuntu, Xubuntu), dependiendo del entorno de escritorio que estos utilicen por defecto (GNOME, KDE). Todas las interfaces están implementadas utilizando el lenguaje de programación Python, pero utilizan librerías gráficas diferentes, GTK para los productos que utilizan GNOME y QT para los que usan KDE.

En el desarrollo de Ubiquity es utilizado como back-end[3] para muchas de sus funcionalidades, el instalador de la distribución de GNU/Linux Debian conocido como “d-i”.

---

[3] Término utilizado en el diseño de software.

Ubiquity puede ser iniciado directamente al seleccionar la opción en el GRUB[4] de Instalar Ubuntu o puede ser iniciado manualmente desde la opción Live CD[5].

Entre las funcionalidades de Ubiquity se destacan las siguientes:

Selección de idioma y distribución de teclado.

La posibilidad de seleccionar el idioma a utilizar durante el proceso de instalación y por defecto en el sistema instalado, permite al usuario interactuar con el instalador con una mayor facilidad. También es posible seleccionar la distribución de teclado a utilizar, evitando que el usuario tenga que realizar todas estas configuraciones al iniciar el sistema operativo instalado.

Gestión de particiones.

Este es uno de los procesos más complicados a la hora de instalar un sistema operativo ya que de no tenerse los conocimientos necesarios podría ocasionar graves daños en el disco duro o pérdida de información. Para facilitar el manejo de las particiones y hacer más intuitivos cada uno de los pasos a realizar, Ubiquity brinda varias opciones para el particionado del disco duro. Una de ellas es la de que instalar Ubuntu junto a otros sistemas operativos buscando de forma automática algún espacio libre donde pueda ser instalado. En caso de no existir un espacio libre que se corresponda con los requerimientos de tamaño del sistema, se cambiarán los tamaños de las particiones necesarias sin perder los datos de las mismas.

También es posible utilizar todo el disco duro para la instalación, advirtiendo que está opción borrará todos los datos del mismo. Ambas opciones muestran una gráfica en forma de barra que permite visualizar como quedará particionado el disco duro después de la instalación. Al utilizar todo el espacio del disco se crea una partición para el sistema y otra que será utilizada como partición de intercambio o swap.

La posibilidad de particionar el disco duro manualmente es recomendada solamente para usuarios con conocimientos en el tema.

---

[4] Administrador o gestor de arranque múltiple.

[5] Sistema operativo que puede ejecutarse desde un CD sin ser instalado.

Las opciones que realizan el particionado de forma automática, utilizan ext4[6] como sistema de archivos.

Instalación del sistema.

El proceso de instalación de Ubuntu se basa en la descompresión de un sistema comprimido mediante *squashfs*[7] y la instalación de paquetes adicionales. Un pequeño repositorio contiene los paquetes *.deb*[8] necesarios para el sistema, los cuales son instalados utilizando *APT*[9].

Anaconda.

Instalador utilizado por varias distribuciones de GNU/Linux entre las que se destacan Fedora, Red Hat Linux, CentOS. El estudio se realizó sobre la versión de Anaconda para Fedora 10 y CentOS 5.

Desarrollo y funcionamiento.

Anaconda ha sido desarrollado utilizando el lenguaje de programación Python y la librería gráfica GTK. Es una solución que permite instalar y configurar sistemas operativos con base Red Hat. Este instalador es ejecutado directamente al iniciar un CD o DVD de instalación.

Gestión de particiones.

Anaconda posibilita la configuración del particionado de los discos duros a través de opciones que incluyen configuraciones por defecto, utilización del espacio libre o la gestión manual de las particiones. Las configuraciones por defecto son creadas borrando todas las particiones o solamente las particiones linux.

---

[6] Sistema de archivos con registro por diario.

[7] Sistema de archivos comprimido de solo lectura para Linux.

[8] Extensión del formato de paquetes de software de Debian y derivadas.

[9] Advanced Packaging Tool (Herramienta Avanzada de Empaquetado).

Este instalador utiliza *libparted*[10] y *pyparted*[11] para gestionar las particiones.

Personalización de la instalación.

Anaconda incluye una página del asistente, en la cual se muestran las aplicaciones que se podrían incluir en la instalación a partir de un repositorio de paquetes .rpm[12]. Estos paquetes están incluidos en el disco pero también puede utilizarse el repositorio en Internet de la distribución correspondiente. Además se le permite al usuario adicionar otros repositorios. En caso de utilizarse algún repositorio en Internet, se muestra un diálogo para la configuración de la red y las opciones de Proxy en caso de que sea necesario usar uno. Para la instalación de los paquetes es usado *YUM*[13].

Configuración post-instalación.

Una vez instalado el sistema, al ser iniciado por primera vez, Anaconda permite realizar una serie de configuraciones básica necesarias para iniciar sesión. En este momento se le permite al usuario crear la cuenta que utilizará en el sistema. Además cuenta con las opciones de configurar la autenticación en servidores con este tipo de servicio, en caso de ser necesario.

Yast.

OpenSUSE cuenta con uno de los sistemas de instalación más completos, con una gran cantidad de funcionalidades y opciones de configuración.

Desarrollo y funcionamiento.

Yast es desarrollado principalmente en los lenguajes de programación Perl y C/C++ e incluye una interfaz que utiliza la librería gráfica QT. Un estilo personalizado para el asistente de instalación, en un archivo *.qss*[14], crea un entorno más amigable para los usuarios.

---

[10] Librería para la gestión de discos duros y particiones.

[11] Módulo de Python para el trabajo con la librería libparted.

[12] Extensión del formato de paquetes de software de Red Hat y derivadas.

[13] Herramienta de gestión de paquetes para sistemas Linux.

[14] Extensión del formato de archivos de estilo de QT.

Reconocimiento y configuración de hardware.

Una de sus funcionalidades es la detección y prueba de los dispositivos de hardware que son de interés para el instalador y el sistema, entre los que se encuentran los dispositivos USB, los discos duros, teclado, entre otros. Tener el control sobre el funcionamiento de los dispositivos es de suma importancia, ya que permite brindar opciones en el proceso de instalación en dependencia del hardware que se esté utilizando.

En el caso de Yast, probar el correcto funcionamiento de los dispositivos USB es crítico para poder incluir funcionalidades como la de incluir productos adicionales desde medios externos. La prueba de los controladores de disco duro permite obtener posteriormente información sobre los mismos para utilizarla en el proceso de instalación. Además son detectadas las interfaces de red y permite configurarlas para poder utilizar repositorios de OpenSUSE en los procesos de instalación o actualización del sistema.

Gestión de particiones.

El manejo de las particiones en Yast incluye varias opciones, que en su mayoría requieren tener conocimientos sobre este tipo de tareas. Inicialmente permite optar entre realizar un particionado basado en particiones o en LVM[15]. A partir de esta selección se genera una propuesta de particionado que incluye la partición raíz del sistema y la de intercambio o swap. El sistema de archivo utilizado por defecto es ext4. En caso de que se desee gestionar las particiones manualmente, recomendado por Yast para expertos, se muestra un árbol con los discos y particiones existentes.

Instalación personalizada.

Yast permite seleccionar entre varios entornos de escritorio el que será instalado con el sistema. A partir de esta selección se instalarán desde el repositorio en el disco de instalación todos los paquetes necesarios. En un árbol se muestran todos los paquetes del repositorio organizados por categoría, los cuales podrán ser seleccionados para instalar. Los que sean indispensables para el funcionamiento del sistema son

---

[15] Logical Volume Manager, administrador de volúmenes lógicos para el kernel Linux.

seleccionados por defecto y no pueden ser desmarcados. Además están marcados para instalar los asociados al entorno de escritorio seleccionado y algunos que son de interés para los desarrolladores. Junto con la selección de los paquetes se verifica y muestra el espacio que ocupan en el disco y el que queda disponible. En este proceso se pueden utilizar otros repositorios que defina el usuario o los oficiales de OpenSUSE. La instalación de los paquetes *.rpm* es realizada por la interfaz de gestión de paquetes de Yast.

Experiencias y consideraciones.

A partir del estudio realizado y conociendo el funcionamiento de algunos de los instaladores, de las distribuciones GNU/Linux más usadas en el mundo, es posible afirmar que la versión actual del instalador de Nova, no cuenta con varias de las características y funcionalidades más importantes para este tipo de aplicación.

Con el propósito de mejorar el proceso de instalación de Nova, debe considerarse incluir entre las funcionalidades de la versión 2.0 de Serere, la posibilidad de gestionar los discos y sus particiones desde el propio instalador. Además, debe incluirse un gestor de paquetes que permita seleccionar las aplicaciones que serán instaladas junto con el sistema operativo. También es necesario que el instalador se encargue de reconocer el hardware en el cuál se va a realizar la instalación.

Debe tenerse en cuenta la posibilidad reutilización de código y componentes de las versiones anteriores del instalador y de otros instaladores que se ajusten a las condiciones de desarrollo de Serere 2.0.

Capítulo 2: Entorno de desarrollo.

En el estudio del capítulo anterior se detectó la necesidad de desarrollar una nueva versión del instalador de Nova, teniendo en cuenta las características de los instaladores estudiados. Con el propósito de crear un producto lo más eficiente y amigable posible y lograr un proceso de desarrollo eficaz, que asegure la calidad de los resultados, se decidió crear un entorno de desarrollo que lo permita. Para construir un entorno que posibilite lo antes planteado se tuvieron en cuenta varias herramientas y elementos de los cuales a continuación se exponen los motivos de la selección y la forma de uso.

Metodología de desarrollo.

Para el desarrollo de Serere 2.0 se seleccionó Programación Extrema(XP por sus siglas en inglés). XP pertenece al grupo de las metodologías ágiles y ha sido utilizada con éxito en el desarrollo de las versiones anteriores de Serere. En la versión actual del instalador esta metodología aún se ajusta a la misión del mismo.

Arquitectura.

Metáfora: *El instalador detecta información del hardware, gestiona el particionado del disco duro, descomprime el sistema, configura los archivos necesarios para su funcionamiento e instala paquetes adicionales.*

Lenguaje de programación.

Python ha sido el lenguaje principal utilizado en el desarrollo de las distintas versiones de Serere. Esto se debe fundamentalmente a que es un lenguaje potente, con una comunidad de desarrollo muy activa que se revierte en una gran cantidad de módulos y librerías muy bien documentadas, lo que posibilita el uso de estas en la implementación de funcionalidades del instalador. Para el desarrollo de Serere 2.0, Python también es una opción ideal, pues además de las características antes planteadas, es posible reutilizar y mantener con mayor facilidad módulos, clases y funciones de las versiones anteriores.

Gestión del proyecto.

Como en las versiones anteriores de Serere, se utilizó en un inicio la herramienta Gforge para gestionar las tareas, la documentación y el control de versiones de Serere 2.0. Posteriormente, debido a cambios en la infraestructura tecnológica de la facultad en la cual se llevó a cabo el desarrollo, se terminó haciendo uso de la herramienta Redmine para estos propósitos. Para la gestión de reportes se utiliza la herramienta Bugzilla.

- **Gestión de tareas**

  Las tareas pueden ser asignadas, modificadas o eliminadas por el administrador del proyecto. Es posible asignar tareas a un integrante del equipo. Cada tarea es asignada con una descripción de la misma, el responsable, la fecha de inicio, la fecha de fin, el porcentaje completado, tiempo estimado, prioridad, estado y si depende o no de otra tarea. Al asignar una tarea, el responsable de la misma recibe una notificación vía correo electrónico.

- **Gestión de documentación**

  La documentación está organizada por categorías o grupos atendiendo al contenido de los documentos. En el proyecto Serere se almacena la documentación de tipo administrativo, destinada a los usuarios o a los desarrolladores. Por ejemplo, se guardan informes de investigaciones de interés para el equipo de desarrollo, el manual de usuario para el uso de el instalador o los datos sobre el personal del proyecto.

- **Gestión de reportes**

  A través de la herramienta Web Bugzilla, es posible reportar los errores referentes a Serere que ocurran durante el proceso de instalación de Nova, estos reportes son enviados automáticamente vía correo electrónico al equipo de desarrollo del instalador, de esta forma es posible detectar y resolver todos los errores que puedan aparecer.

Control de versiones.

Para el control de versiones se utiliza el servicio de subversion[16] que brinda Gforge. En este servidor se almacenan las distintas versiones de Serere, en el caso de la versión 2.0, los cambios se gestionan y almacenan de la siguiente forma:

Serere 2.0/

- **Branches/**
  Cuando se va a implementar una nueva funcionalidad, desde aquí se trabaja sobre lo que se denomina una rama del desarrollo para no afectar a la línea principal. Al concluir el desarrollo de la funcionalidad y esta pasar las pruebas pertinentes, se añade al denominado tronco del desarrollo.
- **Tags/**
  Se etiquetan las versiones del producto, por ejemplo una versión Beta de Serere.
- **Trunk/**
  Aquí se encuentra la línea principal o tronco del desarrollo.

Plataforma de desarrollo.

Para desarrollar Serere 2.0 fue utilizado el Entorno de Desarrollo Integrado(IDE, por sus siglas en inglés) Eclipse en su versión 3.3.1. Mediante el uso de plug-ins la plataforma Eclipse incluye una gran variedad de funcionalidades utilizadas tanto para la implementación del instalador como para la gestión del desarrollo del mismo. A continuación se describen algunos plug-ins utilizados:

[16] *Software de sistema de control de versiones.*

PyDev

Posibilita el desarrollo con el lenguaje Python, permitiendo utilizar todas las librerías de este. Incluye completamiento de código, reconocimiento de sintaxis, análisis del código, refactorización[17], debugger[18], consola interactiva, etc.

Subclipse

Incluye funcionalidades para el control de versiones utilizando los repositorios de subversion, permitiendo gestionar las versiones de Serere desde la plataforma.

PyUML

Permite construir diagramas UML o generarlos a partir de código Python. También es posible generar código Python a partir de diagramas UML.

Pruebas unitarias.

La implementación de pruebas unitarias permite probar el correcto funcionamiento de cada uno de los módulos por separado. Se deben escribir casos de prueba para cada función de forma que cada uno sea independiente del resto. En el desarrollo de Serere se utiliza el módulo **unittest** de Python tratando de cubrir la mayor cantidad de código posible. Estas pruebas son muy útiles para la integración continua[19].

Integración continua.

Utilizando la aplicación **Hudson** es posible obtener el código fuente de Serere a partir del repositorio de *subversion* y realizar de forma automática todas la pruebas unitarias implementadas y una auditoría del código, generando un informe completo en el cual se pueden consultar reportes de:

- Cobertura. Cantidad del código cubierto por las pruebas unitarias.
- Violaciones. Incongruencias del código y el estándar utilizado.
- Fallo y aceptación de pruebas.

---

[17] *Técnica de ingeniería de software para reestructurar un código fuente*.

[18] *Programa que permite depurar o limpiar los errores en un código fuente.*

[19] Metodología informática.

Este proceso puede ser ejecutado de forma manual o automática, siendo posible configurar la aplicación para que lo realice cada cierto tiempo o cuando se envíe una actualización al repositorio. El resultado de aplicar la herramienta Hudson a una revisión determinada, es lo que determina si el equipo pasa a desarrollar la próxima funcionalidad o debe corregir la actual.

Interacción con el sistema operativo.

Herramientas y librerías.

En el proceso de instalación y configuración de Nova son utilizados algunos programas y librerías en los cuales se apoya Serere para su funcionamiento, muchos de los cuales han sido usados en las versiones anteriores del Instalador.

Una de las herramientas utilizadas más importante es *unsquashfs*. La función de la misma es descomprimir el sistema operativo Nova comprimido en formato *squashfs*. Su uso se resume en la siguiente línea de comandos:

```
# unsquashfs -d DESTINO -i FUENTE
```

FUENTE: Dirección completa del archivo *.squashfs* a descomprimir

DESTINO: Directorio donde será descomprimido el archivo *.squashfs*

Otra herramienta utilizada es os-prober, la misma permite detectar información de los sistema operativos instalados. Los datos obtenidos se muestran de la siguiente forma:

```
# os-prober

 /dev/sda1:Windows Vista:Windows:chain

 /dev/sda2:Ubuntu Lucid 10.04:Ubuntu:linux

 /dev/sda3:Nova Baire 2.0:Nova:linux
```

Solamente son mostrados los dispositivos que tienen algún sistema operativo instalado. Las herramientas y librerías utilizadas por Serere sen muestran en **(Anexo 3.Tabla 1 y 2)**

Archivos de configuración.

Una de las funciones de Serere es la de configurar el sistema operativo Nova previamente instalado a partir de la información recogida durante el proceso de instalación. Para esto el instalador debe interactuar con una serie de archivos de configuración ya sea modificando algunos existentes como creando otros.

El trabajo de manipulación de archivos de configuración se facilita con el uso de Augeas[20]. Esta herramienta permite utilizar y crear lentes para definir la sintaxis de archivos de configuración, posibilitando consultar y modificar los mismos con cierta facilidad. Además, es posible utilizar Augeas en otros sistemas, simplemente indicando el directorio que se utilizará como raíz. Esta funcionalidad es de interés para el instalador, ya que es necesario obtener información de archivos en el sistema de instalación y modificar otros en el sistema instalado. Para ilustrar este proceso se muestra el siguiente ejemplo:

Se desea obtener el valor de la variable SIZE en el archivo de configuración de Serere que se define de la siguiente forma:

SIZE = 2500

Se accede a este valor de la siguiente forma:

```
get CAMINO/SIZE

2500
```

Para modificar el valor de SIZE:

```
set CAMINO/SIZE VALOR
```

CAMINO: Ruta del archivo de configuración.(Ej: /etc/serere.conf)

VALOR: Valor a asignar a la variable SIZE

[20] Herramienta de edición de archivos de configuración.

Augeas cuenta con lentes para archivos utilizados por Serere como fstab, sudores y otros. Ver archivos de configuración con los que interactúa el instalador (**Anexo 3. Tabla 3**)

Capítulo 3: Descripción de la solución.

Para desarrollar Serere 2.0, la solución completa fue dividida en diferentes historias de usuario **(ver Anexo 2)**, las cuales fueron desarrolladas y probadas en un proceso de iteraciones, que permitió lograr una solución con calidad en un corto período de tiempo. Como resultado del desarrollo se obtuvo un asistente con una interfaz amigable, fácil de utilizar y que interactúa con el usuario creando un ambiente sencillo y agradable en el transcurso del proceso de instalación. Para su mejor entendimiento la solución se muestra en dos partes, el desarrollo de la aplicación de instalación y la construcción del sistema operativo sobre el que esta se ejecuta.

Para consultar los diagramas de clases más representativos generados por las tareas de ingeniería aplicadas a cada historia de usuario, ver **Anexo 4**.

Aplicación de instalación.

Historia de usuario 1: Selección de idioma.

En esta vista se le permite al usuario escoger el idioma que prefiere utilizar durante el proceso de instalación y que usará por defecto el sistema instalado. Al seleccionar una opción, todos los elementos que contengan texto en el asistente de instalación serán traducidos al idioma seleccionado y se actualizará el componente que muestra las opciones de distribución de teclado. Esta información será almacenada y utilizada posteriormente para configurar la variable de entorno LANG en los archivos de configuración /etc/default/locale y /etc/environment en el sistema operativo Nova instalado. El componente de Qt utilizado para mostrar las opciones de idioma es un QComboBox.

Para pobrar esta historia de usuario se diseñaron dos escenarios:

Escenario 1. Selección de idioma.

La información se muestra en el asistente en idioma Español. El usuario selecciona la opción de Inglés.

Se espera que la información del asistente sea traducida y mostrada en idioma Inglés. Además, debe actualizarse el componente que muestra las opciones de distribución de teclado, seleccionando la opción “USA”, correspondiente al idioma seleccionado. Debe escribirse en los archivos de configuración correspondientes la variable de entorno LANG asociándole el valor “en_US.UTF-8” referente al código asociado al idioma Inglés.

Los resultados obtenidos se corresponden con lo esperado.

Escenario 2. Selección de idioma.

La información se muestra en el asistente en idioma Inglés. El usuario selecciona la opción de Español.

Se espera que la información del asistente sea traducida y mostrada en idioma Español. Además, debe actualizarse el componente que muestra las opciones de distribución de teclado, seleccionando la opción “Spain”, correspondiente al idioma seleccionado. Debe escribirse en los archivos de configuración correspondientes la variable de entorno LANG asociándole el valor “es_ES.UTF-8” referente al código asociado al idioma Español.

Los resultados obtenidos se corresponden con lo esperado.

Historia de usuario 2: Selección de distribución de teclado.

Las opciones de configuración referentes a la selección de la distribución de teclado y sus variantes, a utilizar durante el proceso de instalación y por defecto en el sistema instalado, se muestran utilizando los componentes QComboBox y QListWidget, en la misma vista de selección de idioma. La selección puede hacerse de forma automática al seleccionar el idioma o manualmente, al desplegar la lista de opciones, que actualiza a su vez la lista de variantes asociadas a la distribución de teclado seleccionada. A partir de la selección de la distribución y la variante de teclado deseadas, se modificara el archivo de configuración /etc/default/console-setup, asignando los valores correspondientes a las opciones “XKBLAYOUT” y “XKBVARIANT” en el sistema instalado. Además, será ejecutado el comando “setxkbmap” con las opciones seleccionadas que serán usadas durante la instalación. En esta vista se incluye un

componente QLineEdit en el cual el usuario puede comprobar si la opción seleccionada se corresponde con la que desea utilizar.

Para probar esta historia de usuario fue diseñado un escenario:

Escenario 1. Selección de distribución de teclado.

El usuario selecciona una distribución de teclado.

Se espera que la lista de variantes se actualice en correspondencia con la distribución seleccionada. Además, debe poder escribirse en el componente visual destinado para esto, utilizando las opciones de teclado deseadas. También deben ser escritos los valores correspondientes en las opciones del archivo de configuración antes mencionado.

Los resultados de esta prueba se comportaron según lo esperado. Este escenario es repetitivo con cada una de las distribuciones de teclado.

Historia de usuario 3: Detección de discos duros.

Este proceso es parte de la vista designada a la detección de hardware. Su función es la de detectar todos los discos duros que estén conectados a la computadora y obtener toda la información de los mismos que pueda ser de interés para el usuario o para el instalador. El módulo pyparted es utilizado para detectar la información de los discos referentes al tamaño, modelo, tabla de particiones y todos los datos de sus particiones.

Cada disco detectado constituye un objeto de Python con un atributo que contiene una lista de todas sus particiones. Cada partición es también un objeto de Python con todos los datos detectados.

Es utilizado os-prober para detectar los sistemas operativos instalados.

En la vista este proceso se identifica por un ícono de un disco duro que se hace visible al concluir la detección. Además, tiene asociado una etiqueta que identifica al proceso y un ícono que muestra el estado del mismo, en espera, en ejecución, terminado o fallido. Sobre el ícono de estado terminado se muestra información sobre la cantidad de discos detectados.

Para probar el correcto funcionamiento de esta historia de usuario se diseñaron los siguientes escenarios de prueba:

Escenario 1. Detección de discos duros.

Se utiliza un equipo con un disco duro interno y otro externo conectado a un puerto USB.

Deben ser detectados dos discos duros. La lista de discos duros debe contener dos objetos de Python referentes a cada disco. Cada objeto debe tener un atributo que diferencie al disco externo del interno. Al pasar el cursor sobre el ícono de estado del proceso debe mostrarse un mensaje con la cantidad de discos igual a dos.

Se obtiene como resultado lo esperado.

Escenario 2. Detección de discos duros.

Se utiliza un equipo con un disco duro interno conectado. El mismo tamaño de 80GB y esta particionado de la siguiente forma:

- Una partición de intercambio o swap de 1GB.
- Una partición con sistema de archivos ext4, con un sistema operativo Ubuntu instalado y un tamaño de 20GB.
- Una partición con sistema de archivos ntfs, con un sistema operativo Windows instalado y un tamaño de 20GB.
- El resto del disco está libre.

Debe detectarse un disco duro y crearse un objeto de Python de tipo Disco con un atributo tamaño igual a 80GB. El mismo debe contener además una lista con tres objetos de tipo Partición y uno de tipo Espacio Libre. Cada partición debe tener un atributo igual al tamaño correspondiente, uno referente al tipo de sistema de archivos siendo identificado como libre el objeto correspondiente. Los objetos de particiones no libres deben contener un atributo que identifica al sistema operativo que contiene.

Los resultados obtenidos están en correspondencia con los resultados esperados.

Historia de usuario 4: Detección de batería.

Este proceso es el encargado de detectar si existe alguna batería conectada al equipo, para esto se utiliza la interfaz de HAL del módulo dbus de Python. En caso de detectar una, se recogerá información de la misma referente a su estado, al nivel de carga. Esta información será utilizada por el instalador para alertar al usuario en caso de no estar conectado el adaptador eléctrico, para evitar que se termine la carga de la batería antes de terminar la instalación.

Durante la ejecución de este proceso se mostrará un ícono de una batería para identificar al proceso en ejecución. El ícono de estado del mismo mostrará si se detectó alguna batería y el nivel de carga de la misma.

Debido a que esta historia de usuario es de poca complejidad se diseño solamente un escenario de prueba:

Escenario 1. Detección de batería.

Se utiliza un equipo que usa como fuente de alimentación una batería. La misma se está descargando y se encuentra cargada en un 80% de su capacidad.

Se espera que sea detectada una batería. Debe crearse un objeto de tipo Batería con un atributo referente al estado de la misma indicando que se está descargando y otro que indique la carga actual igual a 80%. Al pasar el cursor sobre el ícono de estado debe mostrarse una batería cargada el 80%.

Se obtuvieron los resultados esperados.

Historia de usuario 5: Detección de modelo de computadora.

El módulo dbus permite detectar información referente al modelo de computadora que se está utilizando. Esta información es utilizada para personalizar el proceso de instalación y para ser almacenada en el registro de instalación, pudiendo ser utilizada por el equipo de desarrollo de Serere para resolver los errores reportados.

Se identificará este proceso con el ícono de una computadora y mostrará sobre el ícono de estado la información recopilada.

Para probar esta historia de usuario se diseño el escenario de prueba siguiente:

Escenario 1. Detección de modelo de computadora.

Es utilizado un equipo Notebook, marca Fujitsu, modelo LifeBook A6-110.

Se debe crear un objeto de tipo Computadora con los valores antes mencionados como atributos del mismo. Toda esta información debe escribirse en el registro de instalación /var/log/Serere.log. Al pasar el cursor sobre el ícono de estado de este proceso estos datos deben ser mostrados.

El proceso se comportó como se esperaba.

Historia de usuario 6: Detección de monitor.

Utilizando el módulo xrandr es posible detectar las salidas de video y los monitores conectados a las mismas. De cada salida de video puede conocerse todos los modos y resoluciones disponibles. Además, este módulo es utilizado para configurar las salidas de video en tiempo de ejecución, permitiendo ajustar la resolución de todos los monitores conectados a la mejor resolución posible. En conjunto con la herramienta dexconf, es posible generar el archivo de configuración xorg.conf y configurar los modos de video y resolución disponibles.

El ícono de un monitor representa al proceso y se muestra información sobre la cantidad de salidas de video y su estado.

Se diseñó el siguiente escenario de prueba para esta historia de usuario:

Escenario 1. Detección de monitor.

Se utiliza un equipo con tres salidas de video con las siguientes máximas resoluciones soportadas:

- LVDS – Conectado – 1280x800
- VGA – Conectado – 1024x768
- TV – No Conectado

Deben ser reconocidas las tres salidas de video. Al pasar el cursor sobre el ícono de estado debe mostrase tres salidas de video y dos conectadas. Los monitores conectados al las salidas deben configurarse a su máxima resolución soportada.

Los resultados esperados son obtenidos.

Historia de usuario 7: Detección de interfaces de red.

La detección de las interfaces de red se realiza utilizando la interfaz de HAL del módulo dbus para este propósito. La información obtenida en este proceso sería utilizada para posibilitar que el usuario configurase la red del sistema instalado. Debido a que la aplicación NetworkManager facilita este proceso para el usuario, se consideró innecesario que instalador cuente con una vista con este fin. Para no perder el trabajo realizado, se decidió utilizar esta vista para configurar la red del sistema de instalación de Nova en futuras versiones, permitiendo incluir nuevas funcionalidades como las de instalar el sistema y paquetes adicionales desde servidores de red.

Este proceso de detección se identifica con un ícono representativo de una red, el cual se muestra al finalizar el mismo.

A pesar de que la información obtenida no será utilizada en esta versión de Serere, se diseñó el siguiente escenario de prueba:

Escenario 1. Detección de interfaces de red.

Se utiliza un equipo con dos interfaces de red identificadas como:

- eth0 – Interfaz de red cableada.
- wlan – Interfaz de red inalámbrica.

Deben crearse dos objetos de tipo Interfaz con un atributo que identifique el tipo. Al pasar el cursor sobre el ícono de estado correspondiente se debe mostrar una interfaz de red cableada y una inalámbrica.

Se obtuvieron los resultados esperados.

Historia de usuario 8: Detección de impresoras.

En este proceso serán detectadas las impresoras conectadas al equipo. La información obtenida de las mismas será utilizada en futuras versiones para configurar e incluir los manejadores necesarios en el sistema instalado. El ícono de una impresora se muestra al concluir la detección, identificando así al proceso terminado.

Debido a lo antes expresado, solamente se diseño un escenario de prueba:

Escenario 1. Detección de impresoras.

Se conecta una impresora al equipo utilizado.

La impresora conectada debe ser detectada. Al pasar el cursor sobre el ícono de estado de la misma debe mostrase una impresora detectada.

Los resultados obtenidos se corresponden con los esperados.

Historia de usuario 9: Selección de zona horaria.

Esta vista cuenta con un mapa mundial y dos elementos QComboBox, en los que se muestran las distintas regiones y principales ciudades del mundo, con el propósito de que el usuario pueda seleccionar la zona horaria donde se encuentra ubicado. El componente del mapa, para el mismo propósito, de Ubiquity fue reutilizado. Al mismo se le añadió una característica, que al entender del equipo de desarrollo de Serere, mejoraba la visualización del elemento que muestra la hora en el mapa. El usuario puede hacer clic en el mapa para seleccionar su ubicación o puede interactuar directamente con los componentes que muestran la información referente a la región y a la ciudad correspondiente. La zona horaria es configurada en el archivo de configuración del sistema instalado /etc/timezone.

Para probar esta historia de usuario se diseñó un escenario de prueba:

Escenario 1. Selección de zona horaria.

El usuario selecciona en el mapa la zona horaria correspondiente a su ubicación.

Los componentes que contienen la información referente a la región y la ciudad deben actualizarse, mostrando los datos correspondientes a la selección. Deben escribirse los datos de la ubicación en el archivo de configuración en el formato “Región/Ciudad”.

Los resultados obtenidos se corresponden con lo esperado. Este escenario se repite para cada selección.

Historia de usuario 10: Administración de usuario.

Esta vista incluye cinco componentes QLineEntry o entradas de texto, cuatro de los cuales son utilizados para solicitar los datos necesarios para crear un usuario en el sistema instalado. Los datos solicitados son los siguientes:

- Nombre real. Nombre del usuario que usará el sistema.

- Nombre de usuario. Nombre que será utilizado para iniciar sesión en el sistema instalado.

- Contraseña. Utilizada para autenticar al usuario al iniciar sesión. La contraseña debe ser escrita dos veces para evitar errores de escritura. Los componentes destinados a esta información son configurados para no mostrar los datos entrados por el usuario.

La entrada de texto restante está destinada al nombre con el cual el usuario desea identificar a su equipo. Por defecto se sugiere un nombre generado a partir de la información referente al modelo de computadora que se esté usando.

Para la creación del usuario en el sistema instalado, además de los datos antes mencionados, es necesario incluir los grupos de usuario a los que pertenecerá. Estos grupos son definidos en el archivo de configuración del instalador serere.conf y definen los permisos que tendrá el usuario al usar el sistema.

Con toda esta información es creado el usuario con el comando “useradd” con sus respectivas opciones. Además es configurado el nombre del equipo en el archivo /etc/hostname.

Esta vista contiene dos componentes QCheckBox que muestran las siguientes opciones:

- Utilizar la misma contraseña para el usuario root. Utilizando en comando “passwd” es cambiada la contraseña de root por la definida por el usuario.

- Permitir que el usuario creado inicie sesión automáticamente. Esta opción permite que cuando el usuario utilice el sistema, se inicie su sesión de forma automática, sin solicitar contraseña. Para esto se configura GDM a través del archivo de configuración /etc/gdm/gdm.schemas.

Debido a que los datos son validados y la vista no se considera completa hasta que no se introduzca correctamente la información solicitada, quedó solo un escenario para las pruebas.

Escenario 1. Administración de usuario.

El usuario introduce los datos solicitados. El usuario selecciona las opciones de utilizar la misma contraseña para el usuario root y de iniciar sesión automáticamente.

Debe crearse un usuario con los valores introducidos por el usuario que pertenezca a los grupos predefinidos por el equipo de desarrollo de Serere. La contraseña de root debe ser cambiada por la del usuario creado. La sesión del usuario creado debe iniciarse automáticamente. El nombre del equipo debe aparecer configurado en el archivo para este fin, anteriormente mencionado.

Los resultados obtenidos son correctos.

Historia de usuario 11: Selección de modo de instalación.

Para seleccionar en que disco duro el usuario desea realizar la instalación de Nova, el instalador cuenta con una vista que incluye un componente QListWidget, donde muestra todos los discos duros conectados a la computadora. Además, reutiliza un componente de Ubiquity , denominado Barra de Particiones, para mostrar el estado actual del particionado del disco seleccionado y una vista previa de cómo quedará organizado el disco después de la instalación. Esta barra identifica con colores cada sistema de archivos y muestra el nombre, orden y tamaño de cada partición.

Para seleccionar cuanto espacio del disco duro será usado para instalar Nova o que partición se desea utilizar para el sistema, Serere brinda las siguientes opciones:

- Utilizar todo el disco duro. Esta opción indica que será utilizado todo el espacio del disco seleccionado para la instalación. Debido a esto se borraran todas las particiones y se creará una nueva que ocupe todo el disco en la cual se instalará Nova. Se muestra una vista previa.

- Instalar en el espacio libre. Esta opción solo estará habilitada en caso de que exista algún espacio libre y este sea mayor que el necesitado para la instalación. El espacio que ocupará la instalación se define en el archivo de configuración del instalador serere.conf. En caso de existir más de un espacio libre, se utilizará el mayor. Se muestra una vista previa.

- Personalizar el disco duro. Incluye una nueva vista al instalador que permite al usuario organizar el particionado del disco seleccionado de forma manual. No se muestra una vista previa.

Para probar esta historia de usuario se diseñaron los siguientes escenarios:

Escenario 1. Selección de modo de instalación.

El usuario selecciona un disco duro. El disco seleccionado no tiene ningún espacio libre. El usuario selecciona la opción de utilizar todo el disco.

La opción de instalar en el espacio libre debe estar deshabilitada. Se debe mostrar la barra de vista previa ocupada totalmente por Nova.

Los resultados son los esperados.

Escenario 2. Selección de modo de instalación.

El usuario selecciona un disco duro. El disco seleccionado tiene un espacio libre pero este es menor que el necesitado. El usuario selecciona la opción de personalizar el disco.

La opción de instalar en el espacio libre debe estar deshabilitada. No debe mostrase ninguna barra de vista previa. Debe añadirse una vista para el particionado manual.

Los resultados son los esperados.

Escenario 3. Selección de modo de instalación.

El usuario selecciona un disco duro. El disco seleccionado tiene un espacio libre y es mayor que el necesitado.

La opción de instalar en el espacio libre debe estar habilitada. Al seleccionar esta opción debe mostrarse la barra de vista previa con el espacio libre ocupado por Nova.

Los resultados son los esperados.

Escenario 4. Selección de modo de instalación.

El usuario selecciona un disco duro. El disco seleccionado tiene dos espacios libres y los dos son mayores que el necesitado.

La opción de instalar en el espacio libre debe estar habilitada. Al seleccionar esta opción debe mostrarse la barra de vista previa con el mayor de los dos espacios libres ocupado por Nova.

Los resultados son los esperados.

Historia de usuario 12: Administración de particiones.

Esta es la historia de usuario más compleja de la solución. Consiste en una vista que solo se incluirá en el caso de que el usuario seleccione la opción de personalizar el disco duro en la vista correspondiente. La vista de particionado incluye una barra que muestra la organización del disco y un árbol que muestra las características de las particiones del mismo, ambos componentes reutilizados de Ubiquity. Al árbol de particiones se le añadieron dos nuevas columnas para mostrar información sobre los sistemas operativos que existan en las particiones y sobre el espacio usado en las mismas. Para manejar el particionado el usuario interactúa con el árbol, permitiéndosele crear, modificar o eliminar particiones. Podrá crear una nueva partición solamente si se selecciona un espacio libre. Si la partición seleccionada no es un espacio libre, podrá editarla o eliminarla.

Crear nueva partición.

Para crear una nueva partición se muestra un componente QDialog para que el usuario introduzca los datos necesarios, como son, el tamaño, el tipo de sistema de archivos, el tipo de partición (Primaria o Lógica), la ubicación (al inicio o al final), el punto de montaje y la etiqueta de la partición. Se hacen las comprobaciones necesarias para validar el número de particiones primarias y la ubicación de las particiones lógicas.

Editar partición.

Se poden editar los datos de una partición a partir de un QDialog con los campos a modificar. Puede cambiarse el tamaño, el tipo de sistema de archivos, el punto de montaje, la etiqueta y puede seleccionarse si se desea formatear o no la partición. Al cambiar el tamaño de una partición solo se permitirá reducir hasta el espacio utilizado en la misma, evitando así pérdida de datos.

Eliminar partición.

Una partición que no sea un espacio libre puede ser borrada, haciendo clic derecho sobre ella y seleccionando la opción correspondiente. Inmediatamente se actualiza el árbol y se muestra un espacio libre con el tamaño de la partición borrada.

A estas opciones puede accederse haciendo clic derecho sobre la partición o haciendo doble clic sobre la misma para crear o editar en dependencia de la partición seleccionada.

Todos los cambios son actualizados en el árbol y en la barra de particiones.

Se lleva el control sobre los puntos de montaje seleccionados para evitar una duplicación y errores posteriores. La vista se considerará completa cuando se haya indicado alguna partición para ser la raíz del sistema.

Debido a todas las medidas de seguridad y validaciones anteriores solo queda espacio para los siguientes escenarios de prueba:

Escenario 1. Administración de particiones.

El usuario selecciona un espacio libre para crear una nueva partición. Introduce un tamaño para la misma, igual a la mitad del espacio libre. Se selecciona el tipo de partición, el sistema de archivos y se ubica la partición al inicio del espacio libre.

Debe actualizarse el árbol y la barra de particiones y mostrar la nueva partición ocupando desde el inicio hasta la mitad del antiguo espacio libre y dejando es su estado original al resto.

El resultado se comporta como era esperado.

Escenario 2. Administración de particiones.

El usuario selecciona partición para editarla. Mantiene el tamaño para la misma. Se selecciona el mismo sistema de archivos y se edita manualmente la opción del punto de montaje, seleccionando uno que está siendo utilizado.

Debe mostrarse un diálogo que muestre el error de duplicación de punto de montaje. No debe modificarse el árbol ni la barra de particiones.

El resultado se comporta como era esperado.

Historia de usuario 13: Instalación personalizada.

Esta historia de usuario se basa en mostrar en un árbol todas las aplicaciones o grupos de paquetes de un repositorio. El usuario puede interactuar con el árbol para hacer una

selección de los paquetes adicionales que desea instalar. Puede seleccionarse un grupo o directamente los paquetes deseados. Al seleccionar un grupo, los paquetes correspondientes al mismo serán seleccionados. La información de los grupos se encuentra en archivos en formato XML organizados por un identificador, un nombre y una lista de los paquetes que agrupan.

El módulo de YUM para Python permite adicionar repositorios de paquetes .rpm e instalar los mismo en la raíz del sistema que se desee, permitiendo realizar este proceso desde un sistema hacia otro.

Debido a la idea de cambiar la base de Nova y consigo el sistema gestor de paquetes se decidió implementar solamente la vista de paquetes y el modulo encargado de su instalación utilizando YUM y paquetes .rpm. En dependencia de la decisión tomada, se mantendrá este módulo o se implementará otro.

El escenario de prueba diseñado se enfoca en la construcción del árbol de paquetes y el la interacción con el mismo.

Escenario 1. Instalación personalizada.

Se utiliza un repositorio local.

Debe construirse un árbol con todos los grupos de aplicaciones existentes en el repositorio. Al desplegar cada grupo deben mostrarse todos los paquetes pertenecientes al mismo. Cuando se seleccione un grupo todos los paquetes contenidos en él deben ser marcados para instalar.

El resultado se obtuvo como se esperaba.

Historia de usuario 14: Resumen de instalación.

El asistente de instalación cuenta con una vista para mostrar al usuario un resumen de los datos que se han recogido en las vistas anteriores y serán usados en la instalación y configuración de Nova. Esta vista se implementó con propósito exclusivamente informativo. Además de recordar las selecciones hechas referentes al idioma, la distribución de teclado, la zona horaria, las opciones de usuario y los puntos de montaje, se hace una advertencia sobre las particiones que contienen sistemas

operativos y las que van a ser formateadas para evitar que el usuario pueda perder datos de sus particiones involuntariamente.

Atendiendo a lo antes planteado fue diseñado un solo escenario de prueba:

Escenario 1. Resumen de instalación.

Se introducen los datos solicitados por el instalador de la forma siguiente:

- Se selecciona el idioma Español.
- Se selecciona la distribución de teclado “Spain”.
- Se selecciona la localización “América/Habana”.
- Se selecciona como raíz del sistema a la partición /dev/sda1
- Se introduce el nombre real y de usuario “Usuario Prueba Serere” y “usuarioserere” respectivamente.

La vista de resumen debe mostrar la información anterior. Debe mostrarse además una advertencia sobre el uso de la partición /dev/sda1, la cual será formateada antes de instalar Nova en ella.

Los resultados fueron correctos.

Sistema operativo.

El proceso de construcción del sistema operativo sobre el cual se ejecutará la aplicación de instalación fue dividido en cuatro pasos:

Instalación de la base.

Debido al cambio de base de Nova a Ubuntu se decidió utilizar como sistema operativo base para el instalador la versión más reciente y estable en el momento del desarrollo, Ubuntu 9.10 Karmic Koala.

Se utilizó la herramienta debootstrap que construye automáticamente un sistema operativo mínimo a partir de un repositorio en una dirección local seleccionada. Los pasos seguidos se muestran a continuación:

#sudo mkdir /tmp/SistemaSerere

#sudo debootstrap karmic /tmp/SistemaSerere http://ubuntu.uci.cu/ubuntu

Instalación de herramientas y librerías.

Contando con un sistema funcional se procedió a instalar las herramientas y librerías que necesita el instalador para su funcionamiento. Para la instalación de los paquetes correspondientes se siguieron los siguientes pasos:

- #sudo chroot /tmp/SistemaSerere
- Se configuró el archivo /etc/apt/sources.lst con los repositorios correspondientes.
- Se utilizó el gestor de paquetes **apt** para instalar cada uno de los paquetes necesarios.

Instalación de la aplicación de instalación.

Una vez instaladas todas las dependencias del instalador quedaba solamente instalar la aplicación de instalación. Para esto se implementó un script en Python para colocar todos los archivos en el lugar correspondiente dentro del sistema.

Configuración del sistema operativo.

Para poder iniciar el instalador automáticamente al iniciar el sistema operativo se hicieron las siguientes modificaciones:

- Se modificó en archivo /etc/event.d/tty1 con la orden de ejecutar el script de Bash /bin/bashlogin, permitiendo iniciar la sesión del usuario root al iniciar el sistema.
- En el archivo /root/.bashrc se añadió la instrucción *startx*. Esta modificación permite que al iniciarse la sesión de root se inicie también el servidor X.
- Para ejecutar el instalador se modificó el archivo /etc/X11/xinit/xinitrc con la instrucción *openbox & serere.* De esta forma se iniciará el instalador utilizando Openbox como manejador de ventanas al iniciarse el servidor X.

Ya el sistema está listo para ser iniciado y utilizar el instalador.

Conclusiones

Al concluir el estudio del funcionamiento de los instaladores de las distribuciones más exitosas de GNU/Linux y conocer las deficiencias de la versión Serere 1.3, quedó demostrada la necesidad de implementar un nuevo instalador para Nova.

Para lograr una solución única que pueda ser utilizada por cada una de las líneas de desarrollo de Nova sin afectar el desarrollo de las mismas, se construyó un sistema operativo GNU/Linux sobre el cual se ejecutará dicha solución. El desarrollo del instalador utilizando el lenguaje de programación Python, permitió reutilizar código de versiones anteriores y de otros instaladores. El uso de Qt como librería gráfica permitió obtener una interfaz con características visuales mejoradas con respecto a las versiones que utilizaron Gtk.

La metodología XP contribuyó al desarrollo eficiente de Serere 2.0.

Actualmente la solución es incluida como instalador de la versión 2.1 del sistema operativo GNU/Linux Nova.

Bibliografía

Anaconda team. Anaconda. 12 de Marzo de 2010. http://fedoraproject.org/wiki/Anaconda (Consultado 18 de Marzo de 2010)

McVittie, Simon. Dbus-Python tutorial. 14 de Junio de 2006. http://dbus.freedesktop.org/doc/dbus-python/doc/tutorial.html (Consultado 20 de Diciembre de 2009)

Nokia. PyQt´s Classes. 20 de Enero de 2010. http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/classes.html (Consultado 2 de Febrero de 2010)

RedHat team. Augeas. 23 de Diciembre de 2009. http://augeas.net/ (Consultado 3 de marzo de 2010)

Rempt, Boudewijn. GUI Programming with Python: QT Edition. 12 de Noviembre de 2009. http://www.commandprompt.com/community/pyqt/ (Consultado 22 de Enero de 2010)

Riverbank Computing. Python Bindings for Qt v4. 20 de Enero de 2010. http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/pyqt4ref.html (Consultado 2 de Febrero de 2010)

Watson, Colin. Ubiquity. 17 de Noviembre de 2009. https://wiki.ubuntu.com/Ubiquity (Consultado 20 de Marzo de 2010)

Yast team. Yast. 31 de enero de 2010. http://en.opensuse.org/YaST (Consultado 22 de marzo de 2010)

Zeuthen, David. HAL 0.5.10 Specification. 16 de Mayo de 2007. http://www.marcuscom.com/hal-spec/hal-spec.html (Consultado 20 de Diciembre de 2009)

Anexos

Anexo 1. Vistas de instaladores

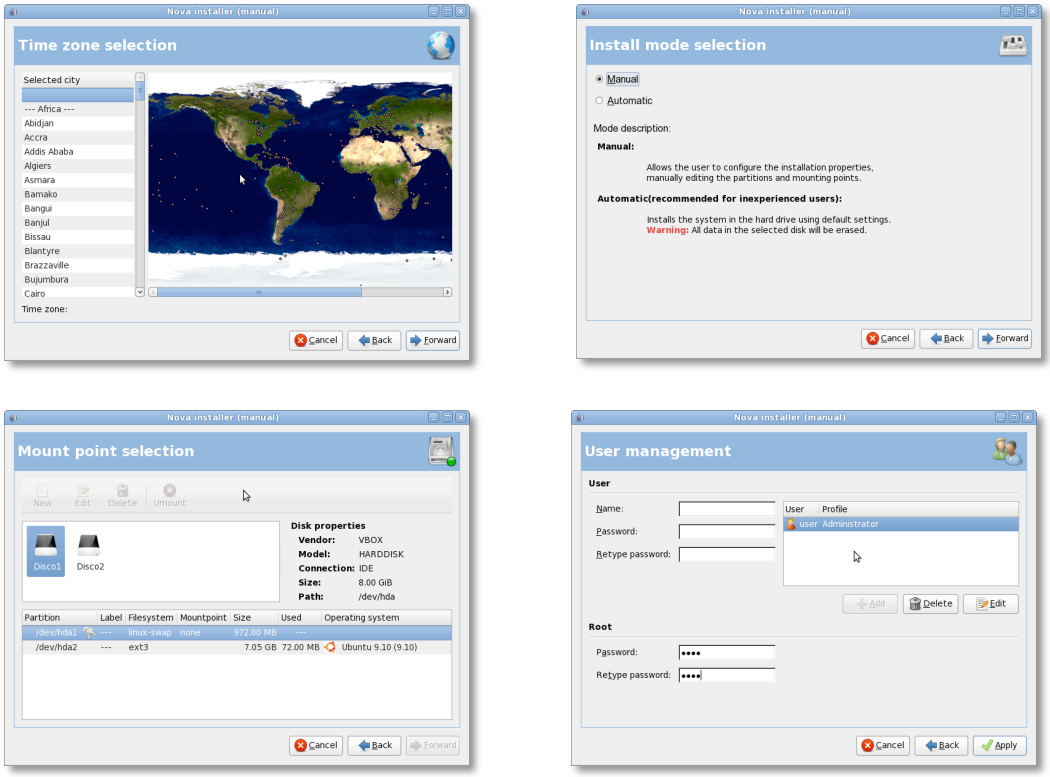

Figura: 1 Vistas de Serere1.3

Figura: 2 Vistas de Ubiquity

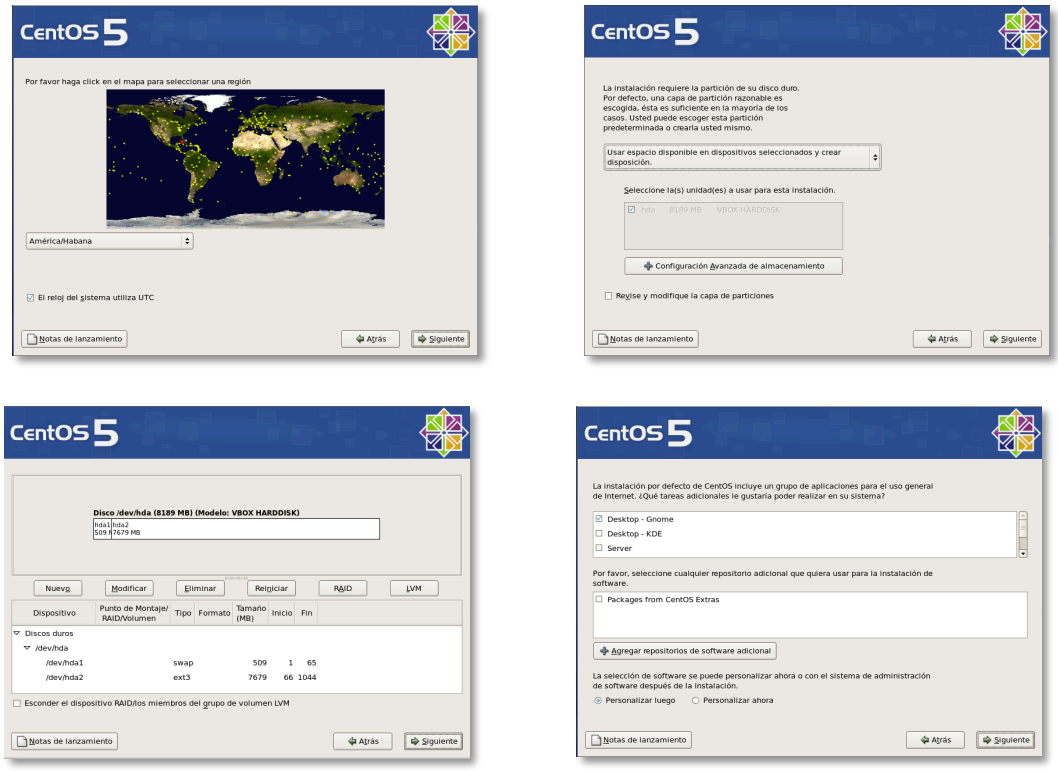

Figura: 3 Vistas de Anaconda

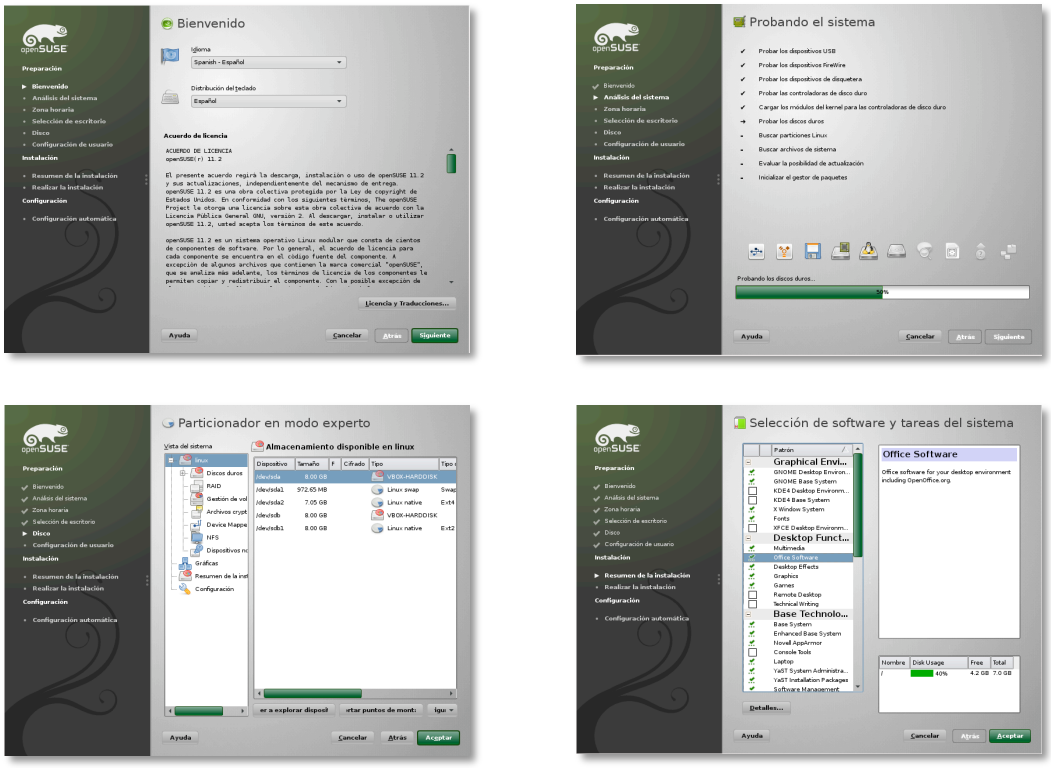

Figura: 4 Vistas de Yast

Anexo 2. Historias de usuario

Historia de usuario	
No: 1	**Nombre:** Selección de idioma
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 1
Descripción: El usuario escoge el idioma que va ha usar el instalador durante el proceso de instalación y el sistema instalado.	
Observaciones: Esta versión solo soporta los idiomas Español e Inglés.	

Historia de usuario 1: Selección de idioma.

Historia de usuario	
No: 2	**Nombre:** Selección de distribución de

	teclado
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 1
Descripción: El usuario escoge la distribución de teclado a usar durante el proceso de instalación y en el sistema instalado.	
Observaciones: Será seleccionada la distribución de teclado correspondiente con el idioma seleccionado.	

Historia de usuario 2: Selección de distribución de teclado.

Historia de usuario	
No: 3	**Nombre:** Detección de discos duros
Usuario: Serere	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 3/5	**Iteración asignada:** 1
Descripción: El instalador detecta todos los discos duros conectados al equipo y se obtiene información de los mismos.	
Observaciones:	

Historia de usuario 3: Detección de discos duros.

Historia de usuario	
No: 4	**Nombre:** Detección de batería
Usuario: Serere	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1/5	**Iteración asignada:** 1
Descripción: El instalador detecta si existe alguna batería conectada al equipo y obtiene información de la misma.	
Observaciones:	

Historia de usuario 4: Detección de batería.

Historia de usuario	
No: 5	**Nombre:** Detección de modelo de computadora
Usuario: Serere	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo

Puntos estimados: 1/5	**Iteración asignada:** 1
Descripción: El instalador detecta información sobre el modelo del equipo.	
Observaciones:	

Historia de usuario 5: Detección de modelo de computadora.

Historia de usuario	
No: 6	**Nombre:** Detección de monitor
Usuario: Serere	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 1
Descripción: El instalador detecta las salidas de video y monitores conectados y configura los modos y resoluciones de los mismos.	
Observaciones:	

Historia de usuario 6: Detección de monitor.

Historia de usuario	
No: 7	**Nombre:** Detección de interfaces de red
Usuario: Serere	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1/5	**Iteración asignada:** 1
Descripción: El instalador detecta las interfaces de red del equipo y obtiene información de las mismas.	
Observaciones: La información obtenida será utilizada en funcionalidades a implementar en próximas versiones.	

Historia de usuario 7: Detección de interfaces de red.

Historia de usuario	
No: 8	**Nombre:** Detección de impresoras
Usuario: Serere	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1/5	**Iteración asignada:** 1
Descripción: El instalador detecta las impresoras conectadas al equipo.	
Observaciones: La información obtenida será utilizada en funcionalidades a implementar en próximas versiones.	

Historia de usuario 8: Detección de impresoras.

Historia de usuario	
No: 9	**Nombre:** Selección de zona horaria
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 2
Descripción: El usuario selecciona la zona horaria de su ubicación a partir de la región o la ciudad donde se encuentra.	
Observaciones: Solamente se destacan las ciudades más importantes de cada región.	

Historia de usuario 9: Selección de zona horaria.

Historia de usuario	
No: 10	**Nombre:** Administración de usuario
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 1
Descripción: Se crea el usuario por defecto en el sistema y se configuran opciones para el uso del mismo.	
Observaciones:	

Historia de usuario 10: Administración de usuario.

Historia de usuario	
No: 11	**Nombre:** Selección de modo de instalación
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 3/5	**Iteración asignada:** 2
Descripción: El usuario selecciona el modo en que desea instalar Nova y el uso que le va a dar a sus discos duros para este propósito.	
Observaciones:	

Historia de usuario 11: Selección de modo de instalación.

Historia de usuario	
No: 12	**Nombre:** Administración de particiones
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto

Puntos estimados: 5/5	**Iteración asignada:** 2
Descripción: El usuario crea, edita y elimina particiones del disco duro seleccionado.	
Observaciones:	

Historia de usuario 12: Administración de particiones.

Historia de usuario	
No: 13	**Nombre:** Instalación personalizada
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 4/5	**Iteración asignada:** 3
Descripción: El usuario selecciona los paquetes adicionales que va a instalar al sistema.	
Observaciones:	

Historia de usuario 13: Instalación personalizada.

Historia de usuario	
No: 14	**Nombre:** Resumen de instalación
Usuario: Usuario	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2/5	**Iteración asignada:** 3
Descripción: El instalador muestra un resumen de toda la información recogida en el asistente.	
Observaciones:	

Historia de usuario 14: Resumen de instalación.

Anexo 3. Interacción con el sistema operativo.

Nombre	Versión	Propósito
unsquashfs	4.0	Descompresión de archivos con formato squashfs.
os-prober		Detección de sistemas operativos instalados.
Grub	0.97	Gestor de arranque.
Bash	3.2.48	Intérprete de comandos.
Python	2.5	Intérprete del lenguaje Python.

Tabla: 1 Programas y herramientas

Nombre	Versión	Propósito
libparted	1.8	Gestión de discos y particiones.
libaugeas	1.8	Interacción con archivos de configuración.

Tabla: 2 Librerías

Dirección	Creado/Modificado	Propósito
/etc/fstab	C	Configuración del montaje de los dispositivos.
/etc/X11/xorg.conf	C	Configuración del servidor X.
/etc/gdm/gdm.schemas	M	Configuración de la ventana de entrada del sistema.
/etc/sudores	M	Configuración de permisos de usuario.
/boot/grub/menu.lst	C	Configuración del gestor de arranque Grub.
/etc/passwd	M	

/etc/shadow	M	Enmascaramiento de los datos de “/etc/passwd”.
/etc/default/locale	M	Configuración de variables de entorno referentes al idioma.
/etc/environment	M	Configuración de variables de entorno.
/etc/default/console-setup	M	Configuración de opciones de teclado.

Tabla: 3 Archivos de configuración

Anexo 4. Diagramas de clases.

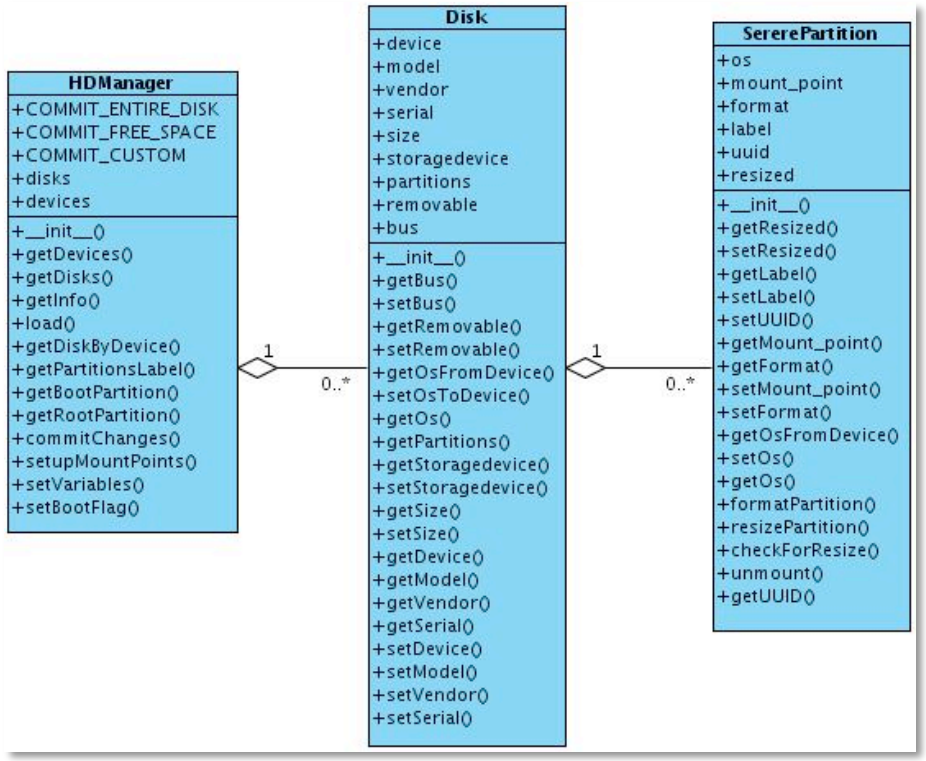

Diagrama de clases 1: Modelado para el trabajo con los discos duros

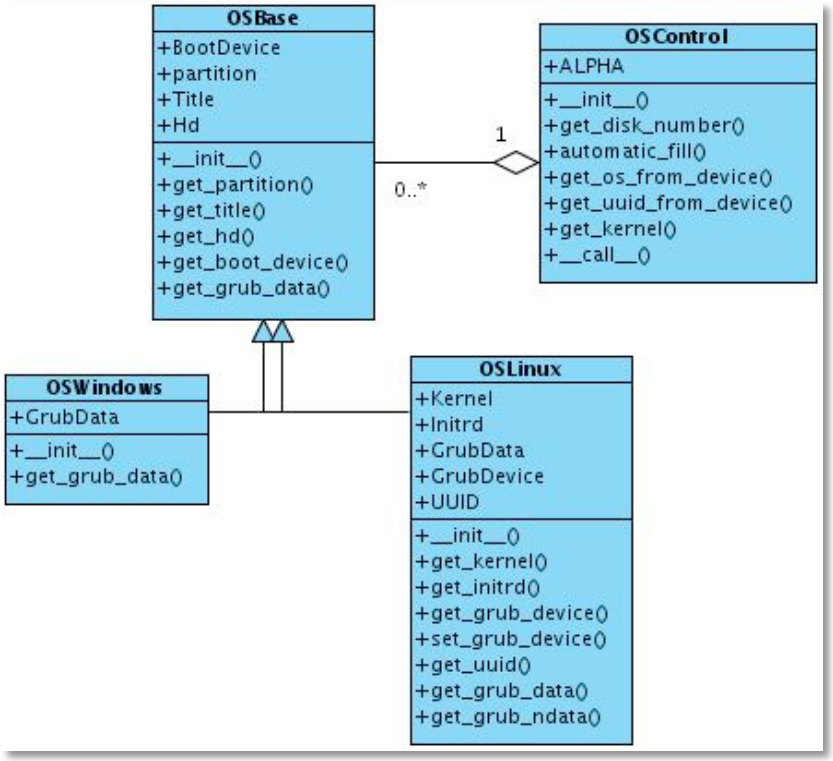

Diagrama de clases 2: Modelado de sistemas operativos para la configuración de GRUB

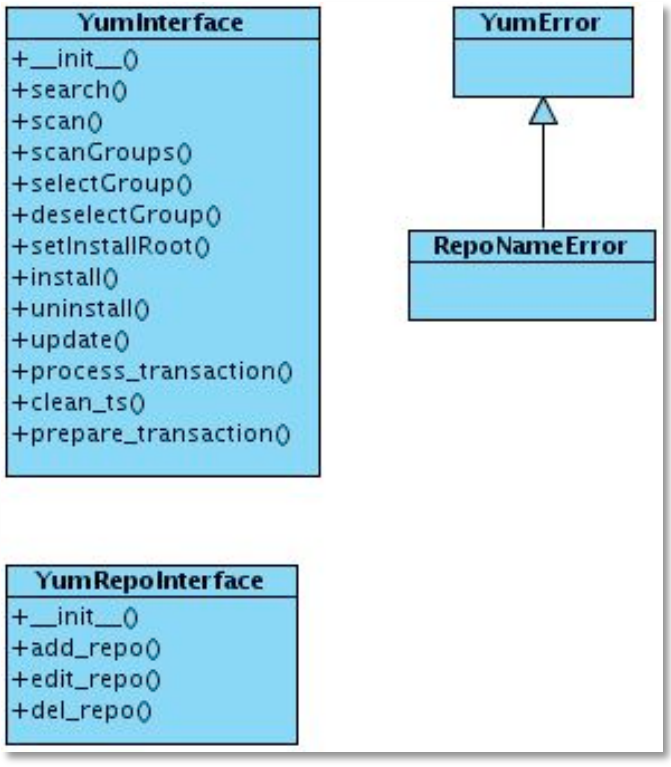

Diagrama de clases 3: Modelado para la instalación de paquetes